Behavior Research Methods & Instrumentation
1982, Vol. 14(3), 329-342

COMPUTER TECHNOLOGY

State notation of behavioral procedures

ARTHUR G. SNAPPER
Western Michigan University, Kalamazoo, Michigan 49008

RONALD M. KADDEN
University of Connecticut School of Medicine, Farmington, Connecticut 06032

and

GEOFFREY B. INGLIS
University of Rochester School of Medicine, Rochester, New York 14642

State notation has been increasingly utilized to describe and implement behavioral procedures since its adaptation for this purpose in 1970. The original version describes states as unique segments of an experimental procedure, accompanied by specified stimulus conditions. Transitions among states are triggered by inputs from the subject, or by the passage of time, and may be the occasion for changing stimulus conditions, recording data, or performing other operations. Extensive usage has suggested a number of possible improvements, and the notation therefore has been expanded and modified. Revisions recently incorporated in state notation increase its power as a descriptive device for effectively communicating the procedural details of reinforcement contingencies.

The use of notation systems in behavioral psychology has a long history. Skinner (1938) found it useful to diagram the difference between operant and respondent conditioning procedures, as did Keller and Schoenfeld (1950). Skinner (1958) also reported on a diagrammatic scheme that he found useful in generating new reinforcement schedules, and Findley (1962) devised diagrams for complex multioperant schedules. The most extensive of the early notation systems was developed by Mechner (1959), and state notation is an outgrowth of Mechner's basic concept, but modified so as to correspond to the notation used in other scientific fields that study sequential processes (see Snapper, Knapp, & Kushner, 1970, for a more complete discussion of the history of state notation).

Although it may take some effort to learn to use a notation system, the consistent use of precise notation has several advantages. The major advantage is that a diagram of the complex contingencies that are present in most experiments can clarify the procedure for the reader. For this reason, state notation has been used to teach behavioral psychology to both undergraduate and graduate students (Lyon & Michael, Note 1). It also has been used to help teachers to understand the complex contingencies of the elementary school classroom (Farris, Note 2). Furthermore, the descriptive clarity of state notation makes it very useful for programming experimental procedures on a digital computer (Snapper & Kadden, 1973).

The use of precise notation has another function that should be of great importance to scientific readers. State notation can avoid misunderstandings resulting from the imprecise descriptions of contingencies that are often found in the procedure section of articles reporting behavioral experiments. This is because state notation makes explicit all sequential properties of reinforcement schedules. An earlier paper (Snapper et al., 1970) reported that two different methods for programming Sidman avoidance have been used by various experimenters, and there is still evidence in the literature that these differences in procedure, which may have important behavioral effects, have not been clearly identified by all researchers who use the schedule.

Another example of the confusion that can occur in procedure sections involves the random-interval (RI) reinforcement schedule. RI schedules have been programmed in two ways, which are likely to produce different behavioral effects. Farmer's (1963) method involves a fixed time period at the end of which the first response to occur has a random probability of reinforcement. A new interval begins after a fixed period of time, regardless of whether or not a reinforcer was delivered. Under Catania and Reynold's (1968) RI,

The authors wish to thank Alan Poling for his helpful comments on an earlier version of this manuscript. This work was supported in part by Research Scientist Development Award MH-70483 from the National Institute of Mental Health, to the senior author. The work of junior author was supported in part by Grants ES-01885, ES-01247, and ES-01248 from the National Institute of Environmental Health Sciences, Grant DA-00623 from the National Institute on Drug Abuse, Grant MH-11752 from the National Institute of Mental Health, and, in part, by a contract with the U.S. Department of Energy at the University of Rochester.

0005-7878/82/030329-14$01.65/0

the first response is reinforced once an interval of random duration has elapsed. The schedule may differentially reinforce longer interresponse times than the former does. Furthermore, both of these methods have two variants according to whether they are programmed "by the clock" or "by the response" (Ferster & Skinner, 1957). It is often difficult or impossible to discover from the description found in a typical procedure section which variant of RI was programmed.

Failures to replicate the findings of others can result from subtle procedural differences of this sort. There is seldom enough information to completely diagram a procedure from the verbal description in an article without contacting the experimenter. For this reason, authors interested in providing enough detail for others to replicate their experiments might do well to either submit with their article a complete state diagram or offer to send a state diagram to interested readers requesting it.

The following description of state notation presents its basic features, using, as illustrations, frequently referenced reinforcement schedules. In the appendix, there is a glossary summarizing the components of the notation.

STATE NOTATION–THE LANGUAGE OF PROCESS CONTROL

State notation was developed to aid the design of sequential switching circuits (Mealy, 1955; Moore, 1956). It has been used to describe sequential processes in which outputs are completely determined by (1) the preceding set of inputs and (2) the current input. For processes of this sort, state notation provides the rules for process control (this includes the procedures of experimental psychology) with minimal amounts of control logic, while completely eliminating indeterminate states (i.e., "race" conditions) (McClusky, 1965).

A modified version of state notation has been adapted to the design of laboratory procedures in which the selection among various possible control sequences is based upon the behavior of experimental subjects. The major advantages of state notation as a process control language are the following: (1) State notation is powerful enough to describe the most complex sequential processes, (2) the simplicity of state notation makes it easy to learn, (3) state notation provides generality across different scientific and industrial process control problems, and (4) since state notation was designed for process control, it is very efficient in terms of necessary control equipment (e.g., permitting as many as 12 concurrent procedures to be controlled simultaneously by a single minicomputer).

Continuous Reinforcement

A basic procedure of operant psychology is regular or continuous reinforcement (CRF). This procedure consists of two states, only one of which can be in effect at any moment. In the first state, reinforcement is not present but is a potential condition. During the second state, a reinforcer (e.g., food) is presented to the subject, usually for a brief duration, by means of a mechanical device.

Figure 1 shows that CRF can be completely described by two mutually exclusive states. Either State 1 is present (represented by the leftmost circle identified by the numeral 1), or State 2 is present, beginning with the onset of reinforcement (labled as "ON SR"). In this procedure, the different states are associated with different stimuli. In more complex procedures, the correspondence between states and stimuli is not always perfect.

The events that cause things to happen ("inputs") in state diagrams are written above the arrow leading from one state to the next. In Figure 1, R1 (a response of Type 1) causes the transition from State 1 to State 2. An input can also produce one or more outputs, which are listed, following a colon, after the input. In Figure 1, R1 initiates reinforcement, indicated by R1:ON SR, if the response occurs when State 1 is present. Four seconds after State 2 is entered, the reinforcement is terminated, shown by 4″:OFF SR. In this case, the end of a 4-sec period constitutes the input and OFF SR is the output.

States also represent response contingencies, in addition to stimulus transitions: in CRF, responses (R1) in State 1 produce reinforcement, but when State 2 is present, responses have no effect (therefore, there is no response transition indicated in State 2).

Transitions, which are considered to be instantaneous, are symbolized by arrows leading from one state to another. Transitions from one state to the next can be initiated by responses (State 1 in Figure 1), or by elapsed time periods, as shown by the 4-sec (symbolized 4″) reinforcement duration of State 2 in CRF. The timing of 4 sec is started on entry into State 2. When the 4-sec period has elapsed, transition back to State 1 occurs.

In the laboratory (and, alas, in real life), reinforcement is not always present or available. For the purposes of an experiment, it is useful to define some discrete event that starts the session and thus presents the opportunity to earn reinforcement, as diagrammed in Figure 2.

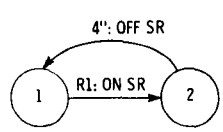

Figure 1. State diagram of CRF.

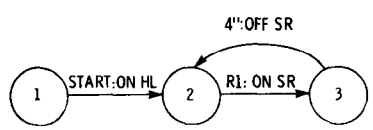

Figure 2. State diagram of CRF including the START input.

The state diagram of Figure 2 consists of three states. State 1 represents the conditions present before the experiment begins. When the START transition occurs (usually an event generated by the experimenter), a stimulus (in this case, the houselight of the experimental chamber) is turned on. In the diagram of Figure 2, the houselight is not affected by responses or reinforcement and therefore is not turned on or off by the response or 4″ transitions. Since no :OFF HOUSELIGHT is included in this diagram, once the experiment begins it continues forever.

Unfortunately, both in the laboratory and in real life, reinforcement opportunities have a limited duration. Figure 3, then, is a complete and rigorous description of a typical laboratory CRF experiment in which 100 reinforcements are available every session.

In Figure 3, a variable (represented by the letter A) is set to zero when the experiment starts. After each reinforcement, A is incremented by 1, as shown by the statement SET A=A+1 on the transition line leading from State 3.

The SET function, introduced in Figure 3 as an output, is borrowed from computer languages (e.g., FORTRAN and BASIC). It is an assignment function that causes the variable to the left of the equal sign to be replaced by the value or expression to the right of the equal sign. Thus, SET A=A+1 is equivalent to saying "replace the current value of A by the old value of A plus 1," or, in other words, add 1 to A. Then a decision is made concerning the next state. At the end of each of the first 99 reinforcements, the transition continues to State 2, permitting more reinforcements. This occurs because A is not yet equal to 100. Therefore, the "ELSE" transition to State 2 will occur. At the end of the 100th reinforcement, transition to State 1 occurs and the houselight is turned off, signaling the end of reinforcement availability.

Multiplication, division, and subtraction operations also may be expressed in the SET function. For instance, SET B=C*D multiples the value of two variables and deposits the results in the variable B. For calculation of rate measures, frequency is divided by time, as in SET R=F/T. Subtraction is illustrated by SET A=B−1.

Decision functions (including the IF statement of Figure 3) are graphically represented by a diamond with the decision criterion written inside it. Since the decision function is initiated by an input, it is therefore a type of output function. The decision function differs from other output functions in that it has two possible output paths. Of course, only one of these paths is taken. The actual path following a decision function depends in any instance upon the validity of the condition specified in the decision function. For example, in Figure 3, if A = 100, then the light is turned off and State 1 is entered. If A is not currently 100 but some other number (1-99 in the example of Figure 3), then the transition to State 2 occurs.

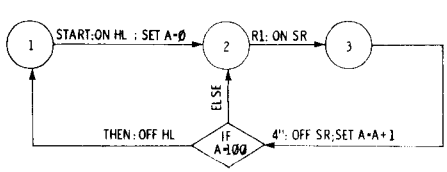

Figure 3. State diagram of CRF including the variable A used to limit session to 100 reinforcements.

In state notation, the comment "THEN" can be drawn on the transition arrow, indicating the path to be taken if the decision criterion is currently valid. The comment "ELSE" indicates the transition that occurs if the conditions are not valid. All decision functions in state notation are binary. That is, either the condition is valid or it is not valid when the decision function is tested. The decision is assumed to be instantaneous, and the decision function should not be confused with a state, since it has no duration.

Figure 3 uses all of the major functions of state notation. Although more complex reinforcement schedules may consist of many sets of contingencies, each complex procedure may be reduced to a series of states, describing the momentary contingencies and stimuli present when each state is in effect. It is important to realize that the state diagram is a description only of potential procedures, but not of the actual outcome of an experiment. For example, in Figure 3, if the subject never responds, reinforcement is never presented; or, if less than 100 reinforcements are earned by the subject, then the houselight continues to shine.

Fixed-Ratio Contingencies

Fixed ratio (FR) is a procedure that is only slightly more complex than CRF. In FR a prespecified number of responses is required for reinforcement. Figure 4 is a state diagram of FR 10. For simplicity, the event terminating the session is not included in this drawing. Notice that the only difference between Figure 4 and Figure 2 is the "10" preceding the transition from State 2 to State 3. The numerical prefix of the response input signifies that the 10th response to occur after entry to State 2 will produce reinforcement and go to State 3. The first nine responses are counted but do not produce transition.

Figure 5 illustrates the fact that the FR procedure may also be diagrammed using variables (e.g., B) and the IF statement.

In this example, each response increments B by 1. The decision function indicates a choice that depends upon the current value of the variable B. If B is not equal to 10, the transition arrow returns to State 2. If B is 10, then B is set back to 0 for the next ratio trial, and reinforcement is presented in State 3. This method of notating FR is cumbersome compared with the method of Figure 4, but it serves to illustrate the logic involved in the numerical counting of response inputs.

Some further rules of state notation are illustrated by the fixed-consecutive-number (FCN) procedure,

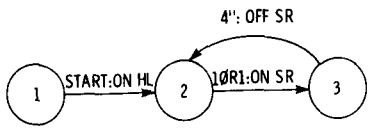

Figure 4. State diagram of FR 10 using response count.

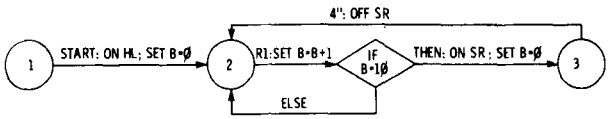

Figure 5. State diagram of FR 10 using the variable B for response count.

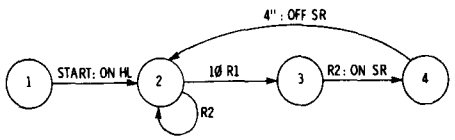

Figure 6. State diagram of FCN using response count.

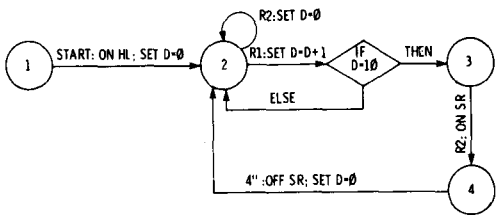

Figure 7. State diagram of FCN using the variable D for response count.

first used by Mechner (1958), as shown in Figure 6.

In this procedure, 10 responses of Type 1 must be emitted consecutively before a response of Type 2 (R2) is reinforced. Responses of Type 2 reset the accumulated count of R1s, and a new set of 10 consecutive R1s must again be emitted without an intervening R2 for transition to State 3. The recycling transition of R2 in State 2 resets the accumulated count of R1s. (Recycling refers to transitions originating and terminating in the same state.) The rule is that the counter of response inputs is reset to zero on state entry. A diagram equivalent to Figure 6, using explicit variables, is shown in Figure 7.

Here, the reset of the accumulated counts of R1 in State 2 is shown explicitly by the expression SET D=0 associated with the recycling transition of R2. Another rule illustrated by the FCN procedure shown in Figures 6 and 7 is that multiple transitions can occur in one state. In State 2, R1 has one effect and R2 has a second effect. In order to avoid conflict among the various possible transitions, there is a basic rule that states that only one input can occur at a time. The input that occurs first will cause its associated transition first. (Of course, in Figures 6 and 7, only the 10th R1 in State 2 will cause a transition to State 3.)

Variable-Ratio Contingencies

Variable ratio (VR) (Ferster & Skinner, 1957) refers to a set of schedules in which the ratio requirement varies after each reinforcement according to some rule. For example, a simple VR schedule with two different ratio values that alternate throughout the session is diagrammed in Figure 8.

For each ratio in the sequence, it is necessary to diagram two successive states: the state containing the ratio value and the reinforcement state. To simplify the diagramming of VR schedules, it is possible to list the sequential values of the VRs and to then set a variable equal to a value from the list. Figure 9 shows a VR consisting of an arithmetic series from 1 to 10 with a mean of 5.5 responses.

Figure 9 introduces four new concepts: (1) the list A, (2) an indexed variable, A(K), (3) SX, and (4) the comment "ALWAYS." The list statement LIST A=3,7... is a shorthand expression equivalent to setting the variable A(0) = 3, A(1) = 7, and so on. It assumes the assignment of 10 variables, each identified by the letter A and an index number from zero to nine. (Conventionally, in this system index values always begin at zero.) Each element of the A list is utilized, in sequential order, under control of the diagram.

The subscripted variable A(K) is used to select values from the list. The subscript K is another variable. In the present example, it can have a value from 0 to 9. When K is 0, A(K) is A(0), so that the expression SET V=A(K) means SET V=3, the value of A(0). When K is 1, SET V=A(K) is equivalent to SET V=A(1), resulting in V's becoming 7, the value of A(1). By controlling the value of K, we control which value from the list, A(0) to A(9), is assigned to the variable V.

SX in Figure 9 is a null transition. Since each decision function has two possible exit transitions, one that will

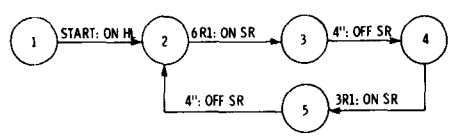

Figure 8. State diagram of VR using consecutive states for each ratio.

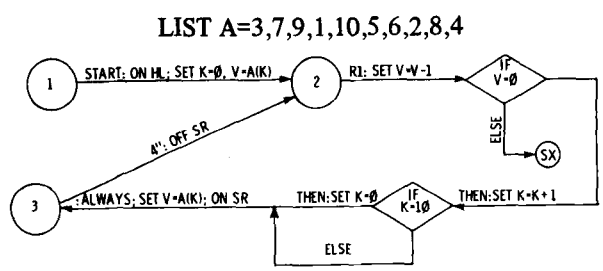

Figure 9. State diagram of VR using the list A(K).

occur if the decision criterion is valid and one that will occur if it is not valid, we need a method for indicating no transition. In the example of Figure 9, if V is not zero, then we merely wish to wait for another R1. We do not need to change states or to cause a transition. This condition can be diagrammed by a transition to SX. In the diagram of VR, it also would have been possible to go to State 2, instead of to SX. However, we will see later that the null transition, to SX, is often useful.

In Figure 9, by a simple LIST statement, we established a list of sequential ratios in which A(0) = 3, A(1) = 7, and so on. On starting the session, the variable V is set to A(K), where K = 0. This means we set V to A(0), or the ratio 3 in this example. Each R1 in State 2 then decreases (decrements) V by 1. The first R1 sets V to 2, the second to 1, and the third to 0. For the first two responses, V is not zero, and therefore each transition is to the null state SX.

Following the third response, V becomes equal to zero, and the value of K is incremented and then checked. If K is 10, it is then set to 0. If K is not 10 or if it has just been reset to 0, then V is set to A(K), the next item of the list. Following completion of the first ratio, K is incremented to 1, and V is set to A(1), the value 7, for the next ratio. By checking the value of K after each ratio, the index of the list can be set back to the first item after the last ratio A(9), or 4, has been used.

"ALWAYS" is used as a comment in Figure 9 to simplify the diagram. We only want to set K to 0 if K is currently at 10. However, we wish to set V = A(K) and turn on SR whether or not K = 10. "ALWAYS" indicates the common path that follows the second decision function, after the decision has been made whether to set K = 0.

To further simplify the notation of variable schedules, two conventions have been established. The first of these is the equivalence between Panels A and B of Figure 10.

Preceding R1 by a variable (Panel A) is equivalent to decrementing the variable upon each occurrence of R1 and then checking the variable for 0. If the variable is 0, then the transition proceeds. Otherwise, the same state remains in effect (the meaning of the null transition to SX).

The second convention is the use of LIST as an output function. Thus, the expression V R1:LIST V=A(K) is defined to be equivalent to Figure 11.

The LIST function is equivalent to the SET and IF functions of Figure 11.

Using these two conventions, VR reduces to the diagram of Figure 12. The sequence of numbers in List A specifies the sequence of particular ratios to be used. Any variable letter can be used in place of V, A, or K.

Interval Schedules

Interval schedules may be programmed in a manner analogous to ratio schedules. For example, consider

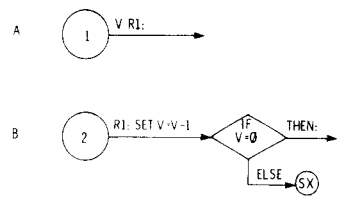

Figure 10. State diagram showing the equivalence of the variable counter V R1 in Panel A with the explicit use of the variable V in Panel B.

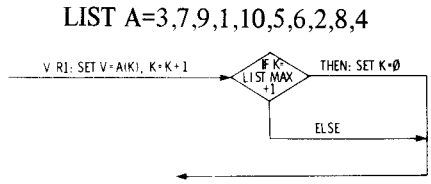

Figure 11. State diagram of equivalence of LIST output function to the explicit use of variables.

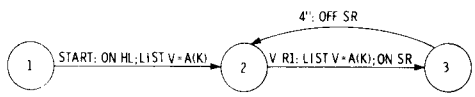

Figure 12. State diagram of VR using the LIST output function and the variable V as a response counter.

Figure 13, which contains the diagram of a 30-sec fixed interval (FI 30-sec) schedule, programmed "by the response" (Ferster & Skinner, 1957).

FI 30 sec imposes a delay period of at least 30 sec after a reinforcement before another reinforcement becomes available. Responses occurring in State 2, before the 30-sec period has elapsed, have no effect. A variant of the FI schedule, in which reponses during the delay period have an effect, is variously called differential reinforcement of low rate (DRL; Ferster & Skinner, 1957) or interresponse time greater than t (IRT > t; Zeiler, 1977) and is diagrammed in Figure 14.

In DRL, responses that occur in State 2, before the 30 sec have elapsed, reset the 30-sec timer, as shown by the recycling R1 transition in State 2. This transition requires the subject to wait at least 30 sec after reinforcement or after an unreinforced response before reinforcement can be obtained.

In some procedures, it is useful to identify independent processes and to diagram these as parallel state sets. Since parallel state sets are assumed to operate independently of each other, a convention must be added at this point: the Z pulse. This is an instantaneous logical signal that may be used to synchronize parallel state sets; a Z pulse is generated as an output in one state set and serves as an input in another state set. An example of this is shown in Figure 15, which illustrates a second variant of the FI schedule, FI "by the clock" (Schoenfeld, Cumming, & Hearst, 1956).

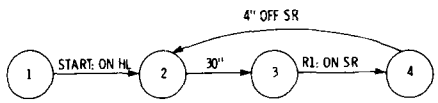

Figure 13. State diagram of FI 30 sec programmed by the response.

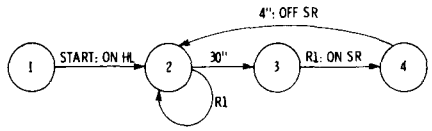

Figure 14. State diagram of DRL 30 sec.

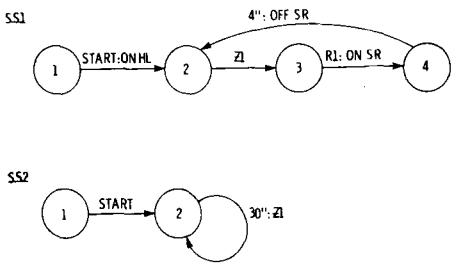

Figure 15. State diagram of FI programmed by the clock.

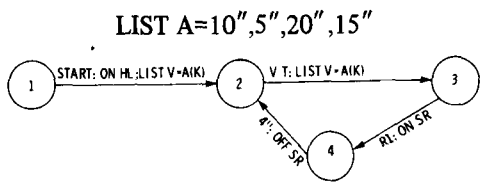

Figure 16. State diagram of VI programmed by the response.

In this variant of FI, the first 30 sec interval begins at the start of the session. Thirty seconds later and every 30 sec thereafter, a Z1 internal synchronizing pulse is emitted.

If State Set 1 (SS1) is in State 2 when Z1 occurs, then State 3 is entered, making reinforcement available for the next response. This schedule differs from FI "by the response" in that the duration of State 2 of SS1 depends upon the behavior of the subject, as well as upon the length of the FI. If the subject delays its response after entering State 3, then the duration of State 2, following the next reinforcement, could be much shorter than the time between Z pulses.

Variable interval (VI) schedules are easily diagrammed using a notation similar to that used for VR: In the arithmetic VI of Figure 16, the LIST output function is used to establish the value of V.

The current version of state notation assumes a basic clock that provides a continuous stream of brief pulses, at a high rate (every .01 sec, for example), to all state sets. In Figure 16, V T indicates that V is decremented at the basic clock rate whenever State 2 is in effect. When V reaches zero, transition to State 3 occurs. Thus, V T works in a way analogous to V R1 in Figure 11.

Still another method of programming VI or VR schedules involves selection of each interval or ratio on a probabilistic rather than sequential basis. For example, random ratio (RR) (Brandauer, 1958) assigns an equal probability of reinforcement to each response. Figure 17 uses a new decision function that incorporates a probability decision.

WITH p=.02 means that the transition line leading to State 3 will occur randomly with a probability of 1/50, or 2% of the cases in which the R1 transition occurs. Otherwise, with probability of .98, the ELSE transition to SX will occur, since all decision functions have one input transition and two output transitions. Figure 17 illustrates an RR schedule with a ratio of 50 responses/reinforcement on the average. Each response in State 2 has 1 chance in 50 of being reinforced (by producing State 3).

RI schedules have also been investigated (Catania & Reynolds, 1968; Farmer, 1963). The method used by Farmer is illustrated in Figure 18.

In this schedule, once every 5 sec there is an opportunity for reinforcement, with probability of .02. The 5-sec interval is timed by the clock, shown in SS2. If response in State 3 is not reinforced, State 2 is reentered. Thus, if a response is emitted at least once every 5 sec, then reinforcement will be delivered on the average every 250 sec, as computed from the formula T/P, where T is 5 sec and P is .02.

An alternative method for programming RI (Catania & Reynolds, 1968) is shown in Figure 19.

In this variant, every 5 sec there is a probability equal to 2% of entering State 3. In State 3, the first response is reinforced. Once again, the average interreinforcement interval is 250 sec. However, there may be some behavioral differences between these two schedules:

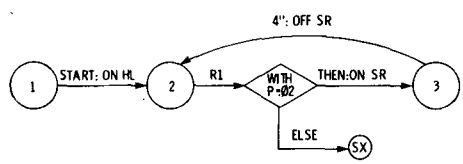

Figure 17. State diagram of RR with probability of response of .02.

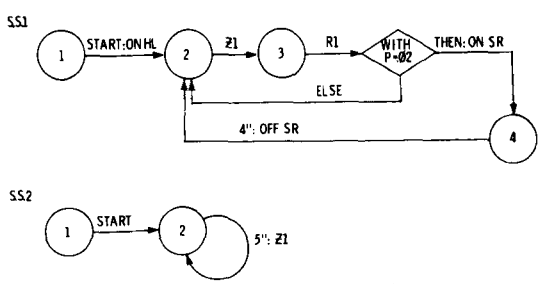

Figure 18. State diagram of RI in which the first response after 5 sec has a .02 probability of reinforcement.

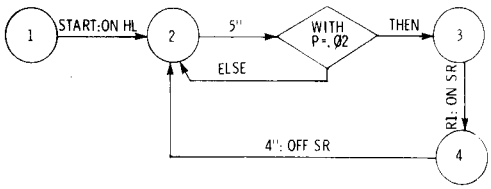

Figure 19. State diagram of RI in which each 5-sec period has .02 probability of setting up reinforcement for the next response.

The Catania-Reynolds method may generate lower response rates, since longer interresponse times have a higher probability of reinforcement.

Randomization

Several methods for diagramming VI and VR schedules have been discussed. A variant of the method utilizing the LIST function has been developed so that a variable may be set equal to a quantity selected randomly from the list without replacement. That is, all items must be selected from the list before any item can be selected again. After all items have been selected once, the function is reset and selection from among the entire set of items can begin again. An example of the use of the RAND function is illustrated in Figure 20.

Figure 20 depicts a discrimination procedure in which a discriminative stimulus (SD) is presented after a variable intertrial interval with a mean interval of 40 sec. On START, and after reinforcement, the intertrial interval begins in State 2 of SS1. The length of the intertrial interval is controlled by the value of the variable timer V T. On entry to State 2, a Z1 operates on SS2, where RAND selects a new value for V from List A, as indicated by RAND V=A.

The RAND function randomly selects one of the seven values from List A and then sets V to this value. In this way, successive sets of seven trials will each have intertrial intervals ranging from 10 to 70 sec, in random order.

The RAND function is particularly valuable in selecting stimuli in random order, for generalization tests. Figure 21 illustrates a procedure in which six stimuli are presented in random order. If the stimulus 1 is presented, then the first R1 after the interval V is reinforced. If one of the other stimuli (2-6) is presented, a response in State 3 is not reinforced but starts a new trial. SS2 randomly selects intertrial intervals from List B and stimuli from List A. Since the stimulus value 1 occurs five times in List A, the number of items in the list is 10. Thus, each stimulus has a probability of 1/10 of being selected in each set of 10 trials, and the stimulus 1 will occur on 5 of those trials. The order of presentation of the stimuli is randomly selected and will differ across successive sets of 10 trials.

Other Complex Schedules

The concept of parallel state sets simplifies diagramming the contingencies controlling session length. For example, consider Sidman avoidance (Sidman, 1953), in which shocks are presented every 2 sec unless a response occurs, and each response produces a 10-sec delay before the next shock. Assume the session is to be terminated after any of the following: the delivery of 500 shocks, the emission of 800 responses, or the passage of 1 h from the beginning of the session. Figure 22 illustrates the procedure, using a parallel state set to control the length of the session.

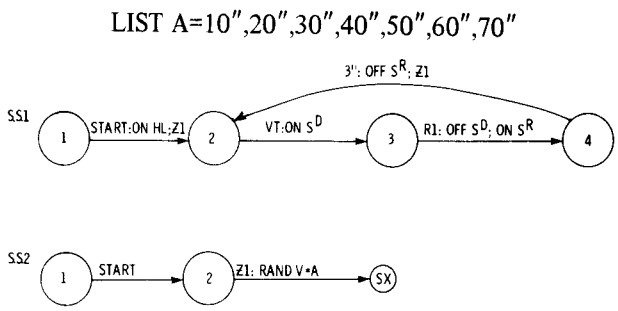

Figure 20. State diagram of a discrimination procedure using the RAND output function to randomly select the stimulus from a list.

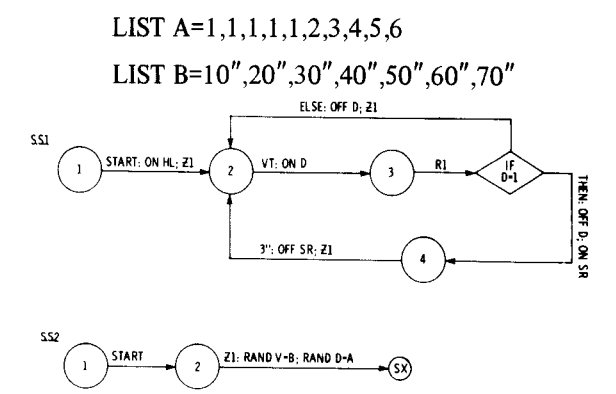

Figure 21. State diagram of a generalization procedure using RAND functions for selecting stimuli and intertrial intervals.

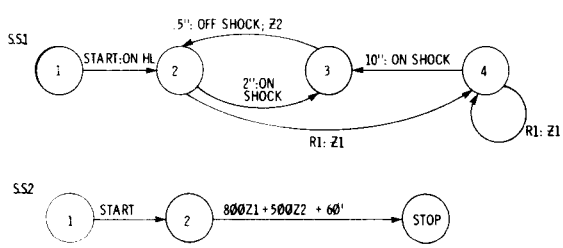

Figure 22. State diagram of Sidman avoidance with session termination after 500 shocks, the emission of 800 responses, or the passage of 1 h.

In SS1, the program alternates between State 2 with no shock and State 3 with a brief inescapable shock every 2 sec. If a response occurs while in State 2, State 4 is entered and is maintained if further responses occur before 10 sec elapse. Note that since R1s have no effect in State 3, the shock is avoidable but inescapable once initiated. After 10 sec elapse in State 4 without a response, shock is delivered. Every R1 in States 2 and 4 of SS1 produces a Z1, and every shock offset produces a Z2. In SS2, the 800th Z1, or the 500th Z2, or the passage of 60 min after the start of the session will cause the session to stop. The plus sign between the inputs in State 2 of SS2 indicates the "logical OR" statement (not the "logical AND").

More complex reinforcement schedules may involve modifications of contingencies according to local response rates. For example, the interlock reinforcement schedule (Berryman & Nevin, 1962; Rider, 1977) involves a ratio requirement that decreases as time from the preceding reinforcement increases. Figure 23 illustrates an interlock schedule in which 50 responses are required for reinforcement at the beginning of each trial.

When a response occurs, or after every 2 sec, the response requirement is decreased by one. At least one response is required for reinforcement, as shown by the final R1 in State 3.

Other schedules with changing requirements for reinforcement have also been investigated. Progressive ratio (Hodos, 1961) is illustrated in Figure 24.

In this schedule, the ratio V is incremented by a fixed amount, 5, upon each reinforcement. In the example of Figure 24, the first ratio is 5, the second is 10, and so on. The session terminates when 15 min elapse without a response.

This schedule needs two variables: V and I. The variable V is set to the current ratio requirement on entry to State 2. Initially it will be 5, then 10, and so on. Each response causes this variable to be decremented by

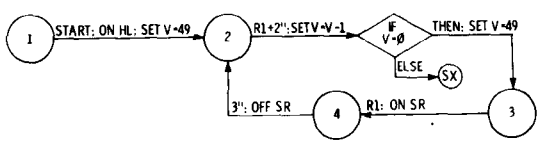

Figure 23. State diagram of interlock schedule of reinforcement.

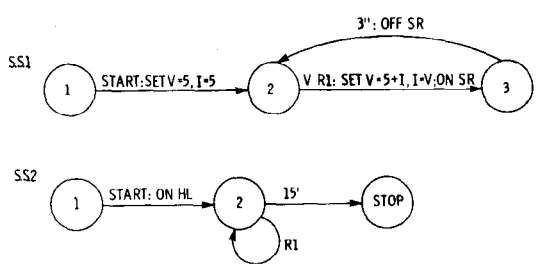

Figure 24. State diagram of progressive ratio.

LIST A=6",12.67",20.17",28.74",38.74",50.74",65.74",80.74"
LIST B=12",25.33",40.33",57.48",77.48",101.48",131.48"

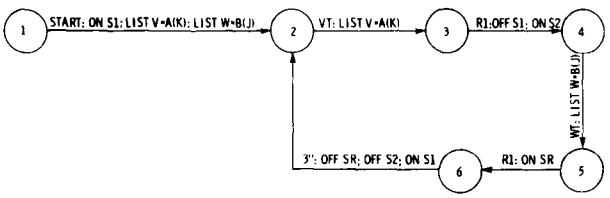

Figure 25. State diagram of chained VI VI.

LIST A=6",12.67",20.17",28.74",38.74",50.74",65.74",80.74"
LIST B=12",25.33",40.33",57.48",77.48",101.48",131.48"

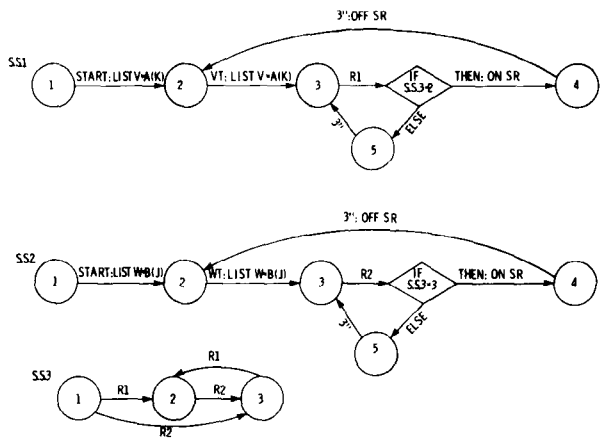

Figure 26. State diagram of concurrent VI VI.

1, through the use of the V R1 statement. The fixed increment, 5, is added to the current ratio requirement for the next trial. The variable I "remembers" the last ratio value initially contained in V. Thus, after the first reinforcement, V is set to 10 (5+I is 5 + 5), and I is also set to 10. After the second reinforcement, V is set to 15 (5+I is 5 + 10) and I to 15 also, and so on. Remember that V R1 decrements the initial value of V as R1s occur, requiring that we remember the initial value of V as a second variable (I).

Chained schedules (Ferster & Skinner, 1957) are easy to diagram. Figure 25 is an illustration of chained VI VI.

The first VI is terminated by a response in State 3. Instead of producing primary reinforcement, the first response after the interval V changes stimulus conditions and initiates a second VI. When a response occurs in State 5, after the interval W, primary reinforcement is delivered and stimulus conditions revert to their initial status.

Concurrent schedules (Ferster & Skinner, 1957) involve two or more different contingencies available for different responses. Figure 26 illustrates concurrent VI VI in which two responses are each reinforced on independent VI schedules.

To reduce reinforcement for switching from one key to the other, a changeover delay (COD) is often pro-

grammed (Catania, 1966). One way to program this delay is to prevent reinforcement for the responses on one key, for a brief period, following responses on the other key. In Figure 26, this is accomplished in SS3, which reflects the most recent response. If an R1 occurred most recently, then SS3 will be in State 2. If an R2 occurred last, SS3 will be in State 3. Responses in State 3 of SS1 and SS2 check the current state of SS3. "IF SS3=2" is equivalent to saying "if State Set 3 is in State 2." If this condition is valid, transition will proceed along the line labeled "THEN." If SS3 is currently in some other state, then transition will proceed along the line labeled "ELSE," and the COD state, State 5, will be entered for 3 sec.

This example also illustrates the fact that there is a temporal sequence to state diagrams. The response in SS1 can check SS3 and find that the preceding response was different only if SS3 has not yet reacted to the current response. The rule is that the response goes to SS1 first, then to SS2, and so forth, in sequential order. Thus, SS3 must temporally follow SS1 for the IF statement to work. If SS1 and SS3 were reversed, then the IF statement would never discover the previous response type.

A frequently used schedule is a concurrent chain, incorporating features of both concurrent and chain procedures. Figure 27 illustrates concurrent VI 1 min VI 1 min chain FI 10 sec FI 30 sec, in which each of two different responses produces a different stimulus condition on independent VI schedules. If either VI (State 2 of SS1 or State 2 of SS3) is completed and its succeeding stimulus condition is presented, then the other VI is suspended until after reinforcement. This is accomplished by producing a Z1 when the stimulus condition of the second component is presented. The Z1 suspends action on the other schedule while retaining the status of its interval timer, until a Z2 is generated after reinforcement. If the VI in SS1 is completed first, then the stimulus 2 is presented and an FI 10-sec schedule is in effect until the reinforcement is delivered. In SS2, in the presence of the stimulus 3, a 30-sec FI is in effect after its preceding VI is completed.

Data Collection

Data collection can be represented by state notation in several different ways. The different types of data collection, which are not mutually exclusive, include (1) simple counts of events or of time durations in different states and (2) calculated data, sorted and treated by means of state table instructions.

Recording of counted data or elapsed time. Collecting simple counts of data or elapsed time is the easiest method to use and often provides the most meaningful and useful information concerning performance of the subject. The method is based on the use of the variable C as a special type of variable.

An example of data acquisition of the simplest sort is shown in Figure 28.

Figure 28 represents a "time allocation" procedure in which the subject can choose between two shock distributions. When the green light ("G") is on in the second state of SS1 and the variable timer VT of SS2 times out, a shock is delivered. On the other hand, if the red light ("R") is on, having been produced by an

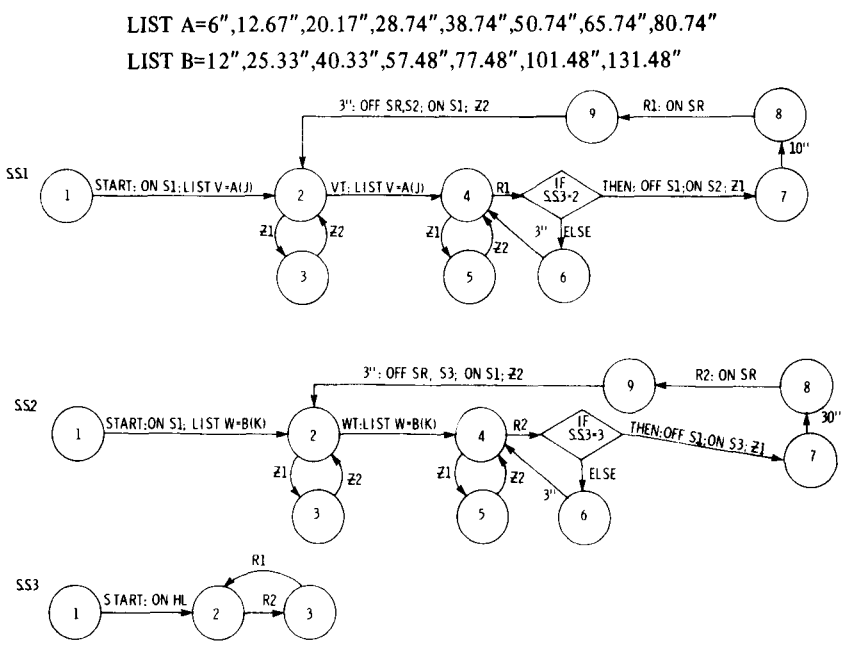

Figure 27. State diagram of concurrent VI VI chain FI FI.

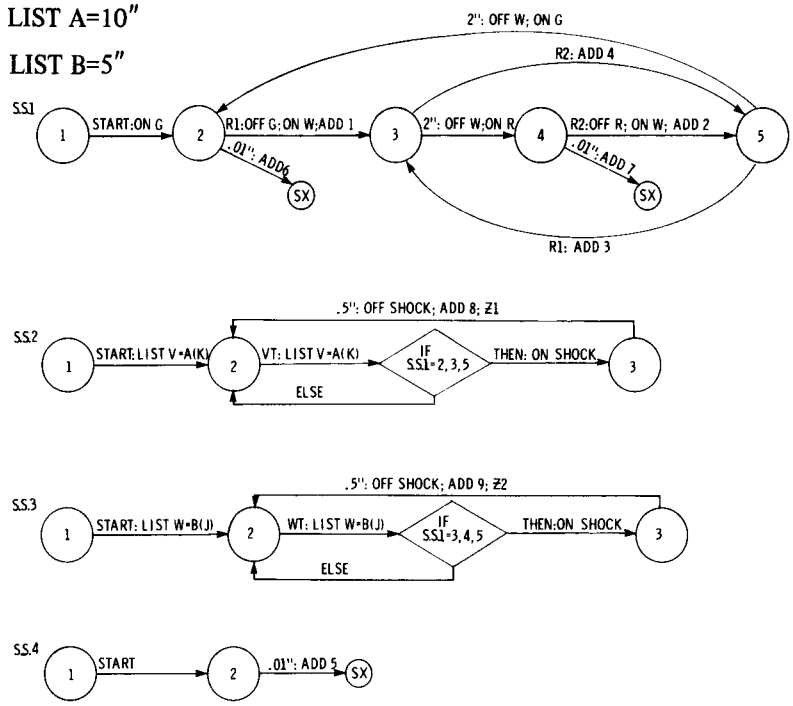

Figure 28. State diagram of a time allocation procedure.

R2, then shocks may be presented by SS3. A 2-sec COD, signaled by a white light ("W"), permits shocks from either distribution in this example. The new function of this diagram is the ADD statement following both responses and times. ADD 1 is a shorthand version of, and completely equivalent to, SET C(1)=C(1)+1. Thus, ADD 1 following each R1 in State 2 of SS1 will increment C1.

Similarly, ADD 6 following each .01 sec spent in State 2 of SS1 will increment C(6), keeping a record of the total time (in .01-sec units) spent in that state. Table 1 describes the contents of each counter.

This recording scheme permits the experimenter to obtain time and response allocation data for the session as a whole. Even if the experimenter were also collecting interevent times, (s)he might want simpler summary data of this sort to describe daily performance. The essential feature illustrated in Figure 28 is that data may be collected and summed over the entire session. On-line data analysis, here, consists merely of collecting the number of responses, shocks, and times elapsed in different stimulus conditions. There is no attempt to look at the fine-grain structure of the behavior or to obtain data that could permit an analysis of the sequential structure of the responding.

Table 1

Counter	Meaning
1	R1s in green
2	R2s in red
3	R1s in COD (State 5)
4	R2s in COD (State 3)
5	Session duration in .01-sec steps
6	Time in green (.01 sec)
7	Time in red (.01 sec)
8	Shocks in green (or white)
9	Shocks in red (or white)

On-line data analysis. A more elaborate analysis of data from the experiment of Figure 28 is easily accomplished. For example, histograms of the latencies between shock and the next changeover response might provide evidence of temporal discriminations of the shock distributions in each component of the schedule. Figure 29 shows the two additional state sets that collect this information.

SS5 records each latency from the delivery of shock by SS2 (Figure 28) to the next occurrence of an R1 that sends SS1 first to State 3 and then to State 4. In State 2 of SS5, a Z1 produced by the end of a shock generated in SS2 will set the variable M to 10. If an R1 occurs in the next 1 sec, then C(M), which in this case is C(10), will be incremented and State 2 will be reentered. If, however, an R1 is not emitted in the first 1 sec of State 3, then M will be incremented to 11. If an R1 occurs in the next 1 sec, then C(11) will be incremented. In this manner, the total number of latencies occurring in 1-sec categories, up to 10 sec, will be accumulated in C(10) to C(19). Latencies greater than 10 sec will be stored in C(20). The IF statement that checks for M's being less than 20 will prevent M's being incremented beyond this value.

SS6 records the R2 latencies following shocks generated by SS3 in C(21) to C(31).

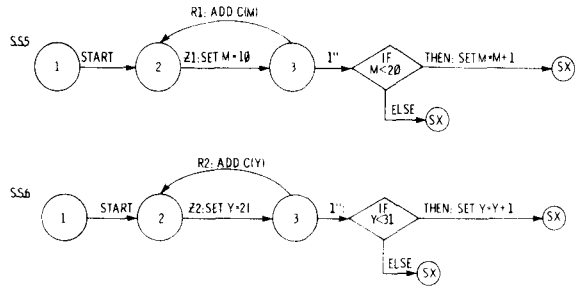

Figure 29. State diagram of a data recording scheme to be added to the time allocation procedure of Figure 28.

STATE TABLE PRIORITIES

State notation, as originally designed (Mealy, 1955; Moore, 1956) required that only one input can occur at a time and that all inputs, transitions, and outputs are instantaneous. These two basic rules guarantee that no conflicts or indeterminate paths ("race" problems in engineering jargon) can occur. The first valid input to occur within a state will cause its associated outputs and transitions. Thus it is possible to predict the outputs and transitions that will occur when a particular input occurs in a particular state.

The theory of sequential machines (McClusky, 1965) deals only with single state sets, whose relationships to the world outside are limited to receiving inputs and sending outputs. Although all sequential procedures, including reinforcement schedules, could be diagrammed by means of a single state set, we have suggested that for clarity, simplicity, and often for reduction in the number of states, it is useful to introduce parallel state sets. The concept of parallel state sets is implicit in the theory of sequential machines, since any input to a sequential machine can be generated as an output by another sequential machine, also described by state notation.

Snapper et al. (1970) proposed explicit construction of parallel state sets that communicate by Z pulses that are produced as an output by one state set and serve as an input for another state set. The explicit consideration of parallel state sets with their associated Z pulses introduces problems of priorities that must be considered in a formal logical system, even though these problems are rarely encountered in programming reinforcement schedules. The problems arise from two rules of state notation. The first rule is that only one input may occur at one moment in time, and the second is that outputs are instantaneous and simultaneous with inputs. Since Z pulses serve as inputs to states (having been generated by responses, time, or other Z inputs), the Z pulses could logically be considered to be simultaneous with the inputs that produced them. Therefore, the basic rule of only one input at a time would be violated. This problem can be clearly illustrated by a single state set.

In Figure 30, an R1 occurring during State 2 produces a Z1 and causes transition to State 3. The Z1, if it occurs in State 2, should cause transition to State 4. To prevent conflict in diagrams of this sort and to retain the rule that only one input can occur at a time, we must introduce another rule: Z pulses operate after the input that produces them has operated on each state set, but before an input of any other type can occur. For instance, a Z1 produced by an R1 will occur before an R2 is operated on. This is Priority Rule 1. This rule clarifies Figure 30: The Z1 is slightly later than and nonsimultaneous with the R1, and therefore the Z1 will occur in State 3 and will cause transition to State 5. The sequence will be as follows: First, R1 occurs in State 2, producing a Z1 and causing transition to State 3; next, the Z1 will cause transition from State 3 to State 5. Although this type of conflict rarely arises in state diagrams, there must be a well-defined rule that tells us what would happen in this case.

The same type of conflict may occur more frequently in cases involving parallel state sets and will be resolved by Priority Rule 1. For example, in Figure 31, both state sets will be in State 2 after a START pulse.

When an R1 occurs, the following sequence of events will be produced: The R1 will produce both Z1 and State 3 in SS1. Next, the R1 will cause transition from State 2 to State 3 in SS2. The Z1 will then cause transition from State 3 to State 5 in SS1, and in SS2 it will produce a Z2 and transition to State 5. Next, the Z2 will cause transition from State 5 to State 6 in SS1 and will have no effect in SS2. Thus, the R1 will produce State 6 in SS1 and State 5 in SS2.

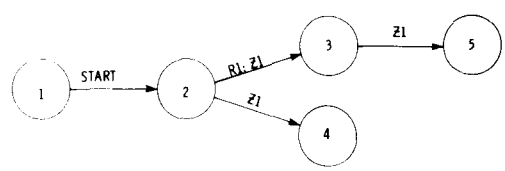

Figure 30. State diagram illustrating Z-pulse priorities.

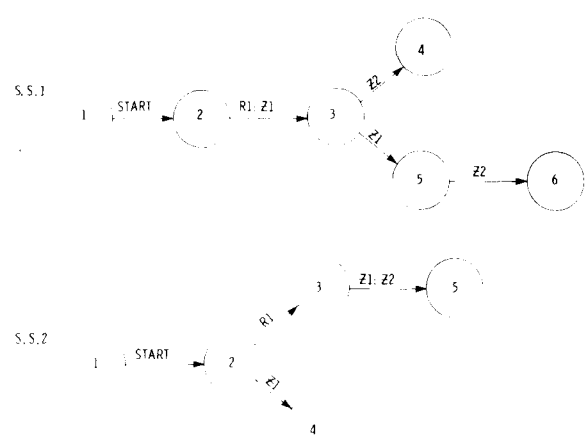

Figure 31. State diagram illustrating Z-pulse priorities.

In general, it is never necessary and seldom useful to use Priority Rule 1 in programming reinforcement schedules. The use of this priority rule will reduce the clarity of the state diagram and make it difficult for others to read.

Furthermore, special care must be exercised when using Z pulses, since infinite loops may be produced by improvident programming. For example, in Figure 32, the start event produces a Z1 and goes to State 2. The Z1 will then produce a Z2 and go to State 3. The Z2 will produce a Z1 and go to State 2. The cycle between States 2 and 3 will continue indefinitely.

Priority conflicts can arise when multiple Z pulses are utilized. In Figure 33, an R1 in State 2 produces both Z1 and Z2.

The Z pulses will be produced following the R1 (Priority Rule 1), but which Z will be effective in State 3? Priority Rule 2 was created to handle conflicts of this sort: When a state has two or more transitions produced by separate Z pulses and when the two or more Z pulses occur on the same input, the transition line in the higher position in the state diagram will occur first. In Priority Rule 2, "higher position" means the uppermost transition line in a state having multiple transitions. In Figure 33, the Z1 transition will take place because it is drawn above the Z2 transition.

In Figure 34, involving parallel state sets, the R1 in SS1 produces Z1 and Z2. In SS2, the Z2 transition will occur, producing State 4, because it is the upper transition line.

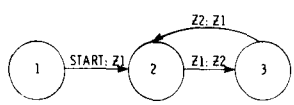

Figure 32. State diagram illustrating an infinite loop.

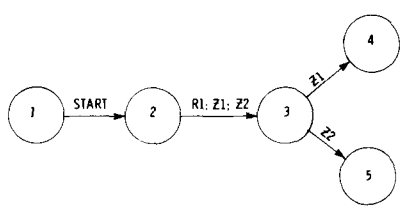

Figure 33. State diagram illustrating priorities among multiple Z pulses in the same transition.

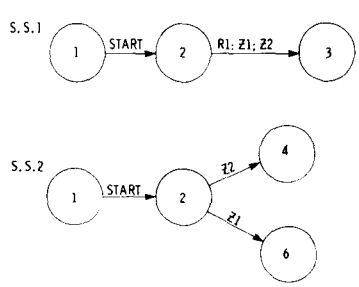

Figure 34. State diagram illustrating priorities of multiple Z pulses across parallel state sets.

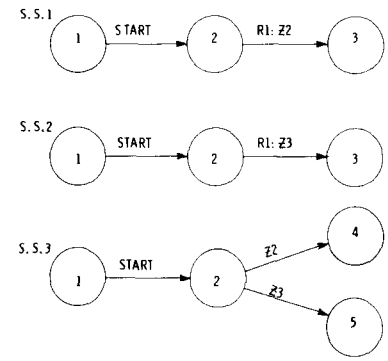

Figure 35. State diagram illustrating priorities of multiple Z pulses produced with different state sets.

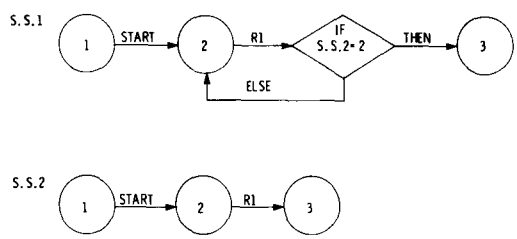

Figure 36. State diagram illustrating priorities in decision functions.

This priority rule must sometimes be invoked when different Z pulses are produced in different state sets by the same input. For example, in Figure 35, the R1 will produce Z2 in SS1 and Z3 in SS2. The result, of course, will be to produce State 4 in SS3. Be sure to check for the occurrence of this sort of conflict when you have a state with two or more Z pulses as inputs. Although Priority Rule 2 permits some programming "tricks" to be used on purpose, it probably should be avoided, since it results in obscure diagrams.

Another priority problem may arise from decision functions that check the current state of another (or the same) state set. Consider Figure 36.

In Figure 36, an R1 in State 2 of SS1 checks the status of SS2 to choose between transition to State 3 and transition to State 2. SS3 advances to State 3 on an R1. If the R1 simultaneously causes transition in the parallel state sets, the decision function of SS1 is indeterminate, since R1 causes SS2 to change states while simultaneously checking the current state of SS2 in the decision function of SS1.

Priority Rule 3 defines the outcome for state diagrams involving this sort of conflict: When an input occurs, it drives state sets sequentially (SS1 first, SS2 next, etc.). Priority Rule 3, then, specifies that R1 will cause transition to State 3 in SS1 when SS2 is in State 2, because the R1 will drive SS1 before SS2. If the order of the state sets were reversed, then the outcome would be reversed.

The R1 in Figure 37 would never cause transition to State 3 of SS2, since SS1 would move from State 2

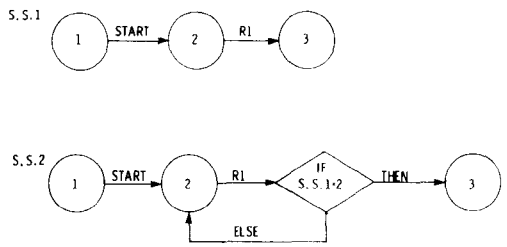

Figure 37. State diagram illustrating priorities in decision functions.

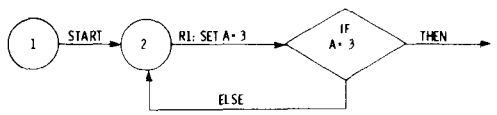

Figure 38. State diagram illustrating priorities in decision functions within a state transition.

to State 3 before the decision function of SS2 could be entered.

A fourth priority rule is needed to define what will happen with IF decision functions that check a variable that is modified on the same transition line in which the decision function is used. Figure 38 illustrates this potential conflict.

Each R1 in State 2 sets A to 3 and checks the current status of A. Priority Rule 4 states that the output and decision functions caused by an input in a state occur in sequential order from left to right. In Figure 38, then, transition will always occur from State 2 to the next state when an R1 occurs in State 2, because A is always set to 3 first. Although this is a trivial and non-useful example, Priority Rule 4 defines the outcome of similar conflicts.

Although the priority rules are arbitrary to some degree, in that other equally usable rules could have been chosen to handle conflicts, the current rules have been used successfully in several versions of state notation.

CONCLUSION

State notation has been utilized, over the past 12 years, by an increasing number of scientists and educators to communicate the details of behavioral procedures. The utility of this system for providing unambiguous descriptions of procedures to computerized control systems gave impetus to its initial development (under the name SKED)[1] and to its subsequent improvement (titled SUPERSKED).[1] Notational clarity has advanced concomitantly with the development of improved control methods. This notational system has proved to be sufficiently flexible to describe a wide variety of procedures, regardless of their complexity.

REFERENCE NOTES

1. Lyon, D. O., & Michael, J. Introducing state notation in university instruction. In A. G. Snapper (Chair), *The use of state notation and the SKED computer system for communication, education, and research.* Symposium presented at the 81st annual meeting of the American Psychological Association, Montreal, 1973.

2. Farris, H. E. *Classroom program booklet: 1975-1976.* Unpublished manuscript, Western Michigan University, 1976.

REFERENCES

Berryman, R., & Nevin, J. A. Interlocking schedules of reinforcement. *Journal of the Experimental Analysis of Behavior*, 1962, 3, 213-223.

Brandauer, C. M. *The effects of uniform probabilities of reinforcement.* Unpublished doctoral dissertation, Columbia University, 1958.

Catania, A. C. Concurrent operants. In W. K. Honig (Ed.), *Operant behavior: Areas of research and application.* New York: Appleton-Century-Crofts, 1966.

Catania, A. C., & Reynolds, G. S. A quantitative analysis of the responding maintained by interval schedules of reinforcement. *Journal of the Experimental Analysis of Behavior*, 1968, 11, 327-383.

Farmer, J. Properties of behavior under random interval reinforcement schedules. *Journal of the Experimental Analysis of Behavior*, 1963, 6, 607-616.

Ferster, C. B., & Skinner, B. F. *Schedules of reinforcement.* Englewood Cliffs, N.J: Prentice-Hall, 1957.

Findley, J. D. An experimental outline for building and exploring multi-operant behavior repertoires. *Journal of the Experimental Analysis of Behavior*, 1962, 5, 113-166.

Hodos, W. Progressive ratio as a measure of reward strength. *Science*, 1961, 134, 943-944.

Keller, F. S., & Schoenfeld, W. N. *Principles of psychology.* New York: Appleton-Century-Crofts, 1950.

McClusky, E. J., Jr. *Introduction to the theory of switching circuits.* New York: McGraw-Hill, 1965.

Mealy, G. H. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 1955, 34, 1045-1079.

Mechner, F. Sequential dependencies of the lengths of consecutive response runs. *Journal of the Experimental Analysis of Behavior*, 1958, 1, 229-233.

Mechner, F. A notation system for description of behavioral processes. *Journal of the Experimental Analysis of Behavior*, 1959, 2, 133-150.

Moore, E. F. Gedanken—Experiments on sequential machines. *Automata Studies, Annals of Mathematical Studies*, 1956, 34, 129-153.

Rider, D. P. Interlocking schedules: The relationship between response and time requirements. *Journal of the Experimental Analysis of Behavior*, 1977, 28, 41-46.

Schoenfeld, W. N., Cumming, W. W., & Hearst, E. On the classification of reinforcement schedules. *Proceedings of the National Academy of Sciences*, 1956, 42, 563-570.

Sidman, M. Two temporal parameters in the maintenance of avoidance behavior by the white rat. *Journal of Comparative and Physiological Psychology*, 1953, 46, 253-261.

Skinner, B. F. *The behavior of organisms.* New York: Appleton-Century-Crofts, 1938.

Skinner, B. F. Diagramming schedules of reinforcement. *Journal of the Experimental Analysis of Behavior*, 1958, 1, 67-68.

Snapper, A. G., & Kadden, R. M. Time-sharing in a small computer based on a behavioral notation system. In B. Weiss (Ed.), *Digital computers in the behavioral laboratory.* New York: Appleton-Century-Crofts, 1973.

Snapper, A. G., Knapp, J. Z., & Kushner, H. K. Mathematical description of schedules of reinforcement. In W. N. Schoenfeld (Ed.), *The theory of reinforcement schedules.* New York: Appleton-Century-Crofts, 1970.

Zeiler, M. Schedules of reinforcement: The controlling variables. In W. K. Honig & J. E. R. Staddon (Eds.), *Handbook of operant behavior.* Englewood Cliffs, N.J: Prentice-Hall, 1977.

NOTE

1. SKED and SUPERSKED are registered trademarks of State Systems, Inc., of Kalamazoo, Michigan.

APPENDIX

Concepts of State Notation

ADD. ADD 1 is equivalent to SET C(1)=C(1)+1. ADD X(3) is equivalent to SET X(3)=X(3)+1.

ALWAYS. A comment to clarify the final common path for decision functions.

C ARRAY. The variable C is unique in that it is used for recording data. (See ADD.) Otherwise, it is similar to any other variable.

DECISION FUNCTION. Specifies what occurs if an algebraic statement is valid at the time of state transition. Every decision function has one input transition and two output transitions. (See "IF," "WITH," "THEN," "ELSE," and "ALWAYS.)

ELSE. A comment used to indicate the path to be taken if the comparison involved in a decision function is not true.

FIXED TIME INPUT. A fixed time input (e.g., 10 sec) is always started or reset on entry to the state in which it is an input. (Contrast with VARIABLE TIME INPUT.)

FIXED COUNT INPUT. A fixed count of responses or Z pulses (e.g., 5R1) is always reset on state entry. (Contrast with VARIABLE COUNT INPUT.)

IF. A decision function that can be used to compare the current value of a variable or the state number of the current state in a state set with another variable or constant and then to select one of two different paths, depending upon the outcome of the comparison.

INDEX. The index, or subscript, of a variable array points to one of the items in the array and is itself a variable.

INPUT. A response, the passage of time after state entry, or a Z pulse. Since inputs are instantaneous, only one input is considered to occur at a time.

LIST. The list is a declaration of values for an array. When the last element of the list is used, the list starts again from the beginning. The LIST function is often used for specifying the sequence of input requirements for variable-ratio or interval schedules, but it may also be used to specify a sequence of values for any variable.

LOGICAL OR. Inputs can be "OR'd," meaning that the completion of any of several input requirements will have the same effect, producing the same transition and the same outputs. The OR symbol is +.

OUTPUT. Any stimulus change or mathematical operation that may occur when an input requirement is completed.

PARALLEL STATE SETS. Subprocesses that are essentially independent can be diagrammed as separate state sets operating in parallel. They may intercommunicate by means of Z pulses.

RAND. The RAND function selects elements from a list at random without replacement until each element of the list has been selected.

RESET INPUT COUNTERS. When a state is entered, all fixed input counter requirements are reset to a specified number. This is true for counters of responses and counters of Z pulses, as well as for counters of the passage of time. However, variable input counters are not reset upon state entry; they are modified only by the SET, LIST, or RAND functions.

SET. An assignment statement used to change the value of a variable to that of a constant, another variable, or an expression.

STATE. A portion of a sequential procedure. A state consists of the contingencies present during some time interval: that is, the designation of the possible transitions that will result when various inputs occur.

STATE SET. A collection of states, and transitions among them, that make up a procedure.

START. An event that can act as an INPUT and will start a session.

STOP. End-of-session command that inactivates all state sets.

SX. SX is used to terminate "null" transitions that neither terminate states nor reset input requirements. Null transitions may, however, generate outputs.

THEN. A comment used to indicate the path to be followed if the comparison of a decision function is true.

TRANSITION. The instantaneous change between two states, triggered by an input. The previously active state is immediately deactivated, and any specified outputs are immediately implemented.

VARIABLE. A quantity whose value changes as a function of algebraic relationships between other quantities. See "SET."

VARIABLE COUNT INPUT. A variable count of responses or Z pulses (e.g., V R1) that always must be explicitly set to an initial value (e.g., SET V=10), since the variable is decremented to zero during the counting process.

VARIABLE TIME INPUT. A variable time (e.g., V T) must always be set to an initial value (e.g., SET V=10″), since the variable is decremented to zero during the timing process.

WITH P. A decision function based on a random probability.

Z PULSE. An internal pulse generated as an output in one state set and used as an input by another state set, to synchronize them.

Priority Rules

Priority Rule 1. Z pulses operate after the input that produces them has operated on each state set, but before an input of any other type can occur.

Priority Rule 2. When a state has two or more transitions produced by different Z pulses and when these two or more Z pulses are produced by the same input, the transition line in the higher position in the state diagram will occur first.

Priority Rule 3. When an input occurs, it drives state sets sequentially (State Set 1 first, State Set 2 next, etc.)

Priority Rule 4. The output and decision functions caused by an input to a state occur in sequential order, as listed from left to right.

(Received for publication March 1, 1982;
accepted March 5, 1982.)